Use of path algebra tools for a unified description of a large class of pull control policies

Maria Di Mascolo, Jean-Marc Bollon

Use of path algebra tools for a unified description of a large class of pull control policies

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0537.R3
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	05-Oct-2009
Complete List of Authors:	Di Mascolo, Maria; Laboratoire G-SCOP Bollon, Jean-Marc
Keywords:	QUEUEING NETWORKS, BATCH PROCESSING, PULL SYSTEMS, CONWIP, DYNAMIC PRODUCTION CONTROL, INVENTORY CONTROL, JIT, KANBAN, MODELLING, PRODUCTION MANAGEMENT
Keywords (user):	(min,) algebra, comparison

International Journal of Production Research
Vol. 00, No. 00, 2008, 1–29

Use of path algebra tools for a unified description of a large class of pull control policies

M. DI MASCOLO* and J-M. BOLLON

G-SCOP laboratory, Grenoble INP, UJF, CNRS-UMR5272 - 46 Avenue Félix Viallet, 38081 Grenoble cedex, France;

(*v1.1 released July 2008*)

Many pull policies can be found in the literature for controlling multistage production/inventory systems. In this paper, we present a framework that enables us to describe the dynamics of a large class of pull control policies, using the same set of canonical functions.

The class of policies we consider here include well known pull policies, like kanban, CONWIP, basestock, generalized kanban, extended kanban, but also many other hybrid policies, and their extensions to systems producing batches.

Each of these policies is characterized by the values of some parameters. These parameters values are calculated thanks to a computational algorithm that relies on the use of path algebra tools, especially $(\min, +)$ algebra tools.

This canonical formulation allows to identify under which values of the control parameters, two different policies have the same dynamics behavior. It also enables to derive methods for evaluating and comparing the performance of several pull control policies, as we illustrate it in the paper.

Keywords: Pull control policies; hybrid policies; (min,+) algebra; queueing systems; batch production; performance comparison.

1. Introduction

This paper deals with make-to-stock pull control policies for multi-stage production/inventory systems. In other words, we consider manufacturing systems which are decomposed into stages. Each stage consists of a manufacturing process (which is a subpart of the studied manufacturing system), and an output buffer (containing semi-finished products). Parts are produced in manufacturing processes in order to maintain the output buffers to given nominal levels. The coordination among the stages is achieved via a pull control policy, i.e., the release of parts into each stage is triggered by real demands for finished products.

*Corresponding author. Email: Maria.Di-Mascolo@g-scop.grenoble-inp.fr

ISSN: 1478-6451 print/ISSN 1478-646X online

DOI: 10.1080/1478645YYxxxxxxx
http://www.informaworld.com

Such pull control policies are characterized by the way the information is transmitted from downstream (i.e. arriving demands) to upstream.

In a general case, two classes of information flows can be distinguished, namely the global and the local information flows (Bonvik 1996). The global information flow links a given stage directly to the arrival process of external demands, whereas the local information flow links the input of a given stage to the output buffer of the stage, and gives an information on the consumption of a part by the downstream stage. Many pull policies can be found in the literature for controlling production/inventory systems. They differ according to the way the information is transmitted upstream. For example, the kanban (Monden 1983) and the CONWIP (CONstant Work In Process) (Spearman *et al.* 1990) use a local transfer of information, whereas the basestock (Lee and Zipkin 1992) uses a global flow of information. There are also some hybrid policies, that combine these traditional policies, and thus local and global information flows. We can quote, among others, the extended kanban (Dallery and Liberopoulos 2000) and the generalized kanban (Buzacott 1989), which both incorporate the kanban and basestock policies, but also some other hybrid policies present in (Bonvik 1996), that combine CONWIP and kanban, or basestock and modified kanban, for example. Other hybrid policies are presented in (Liberopoulos and Dallery 2000). They are variants of well known policies, obtained by adding a local control for the Work In Process of each stage, or by nesting several pull control systems.

Some of the existing pull control policies may have the same behavior, under certain conditions. However, it is often very difficult to know if the behaviors of two policies are identical, since they are often described using different tools, like, for example, queueing networks, Petri nets, sets of equations or control cells. Comparison studies have been conducted those last years. Some of them are motivated by the introduction of a new policy, which is compared to some existing ones, using simulation. See for example (Bonvik 1996), which introduces hybrid policies combining CONWIP and kanban, or basestock and modified kanban, or (Boonlertvanich 2005), which introduces the "extended-CONWIP-kanban" control policy.

Many papers propose a quantitative comparison among some known policies, based on the evaluation of some performance measures, and an optimization of their parameters on some case studies. See for example (Duri *et al.* 2000), which compares the performance of the kanban, basestock and generalized kanban systems, obtained using analytical methods, (Bonvik *et al.* 1996), which compares the performance of the kanban, minimal blocking, basestock, CONWIP, and hybrid kanban/CONWIP control policies using discrete-event simulation, (Geraghty and Heavey 2005), which compares the performance of the kanban, CONWIP, Hybrid Kanban/CONWIP systems using discrete-event simulation, or (Kleijnen and Gaury 2003), which compares four policies (kanban, CONWIP, Hybrid Kanban/CONWIP and generic kanban) using discrete-event simulation, heuristic optimization, risk analysis and bootstrapping. Other papers propose a qualitative comparison using a unified modelling framework. This is the case of (Liberopoulos and Dallery 2000) which proposes a way for defining pull control mechanisms, based on queueing network representation with synchronization stations. The proposed framework helps describing and comparing classical multi-stage production/inventory control policies (base stock, kanban, generalized kanban, extended kanban). It also enables to introduce new control approaches, by adding local mechanisms to control the Work In Process in each stage, or by nesting several pull control systems. Finally, several production control systems that have appeared in the literature are shown to be equivalent to some of the new introduced control systems. In (Liberopoulos and Dallery 2003), this

framework is extended in order to include policies dealing with lot sizing.

In a previous paper (Bollon *et al.* 2004), we proposed a unified framework to describe the dynamics of some pull control policies (basestock, kanban, generalized kanban, and extended kanban). This formulation consists in a function, which is the same for all the studied policies, and whose parameters differ from one policy to another. It thus allows a systematic approach to derive some properties for each policy and to find all identical dynamics between two systems, by comparing the parameters. We used this approach to find some equivalences between the extended kanban and the generalized kanban mechanisms.

In this paper, our first aim is to extend the framework proposed in (Bollon *et al.* 2004), which was limited to the base stock, kanban, generalized kanban, and extended kanban policies. The proposed formulation thus enables now the description of a large class of pull control policies, including all the policies described above and their extensions to systems producing batches, such as those presented in (Liberopoulos and Dallery 2003). Note that, as shown in section 4, many other new policies can also be considered. Our formulation enables us to represent all these policies with the same set of canonical functions, each policy being characterized by the values of some parameters.

Secondly, we derive two algorithms that enable the computation of the formulation's parameters, one for the policies without batch production, and the second for the policies with batch production. These algorithms rely on the use of a shortest path search. They are obtained using path algebra tools, especially $(\min, +)$ algebra tools.

Finally, we show how our formulation can be used to compare the behavior and the performance of pull control policies.

This paper is organized as follows: in section 2, the modelling assumptions used for the systems studied in this paper are presented. In section 3, our formulation is briefly presented and the canonical representation is given for basestock and generalized kanban policies. Some elements about path algebra, that are necessary to the derivation of the new algorithms presented in this paper, are also given. Section 4, shows how this canonical formulation can be extended to more general pull control policies, thanks to the use of path algebra. An efficient computational algorithm that enables us to find the parameters of the canonical representation is derived. In section 5 it is shown how this formulation can be extended to systems producing batches. The corresponding computational algorithm is derived and applied on some pull control policies encountered in the literature. Finally, in section 6, we show how our formulation can be used in order to obtain some properties of pull policies, and how it can be used to calculate performance parameters of a large class of pull control policies.

2. Modelling assumptions

The production/inventory systems that are considered here are divided into N stages. Each stage i consists of an output buffer, denoted by P_i, which contains the finished parts of the stage, and a manufacturing process, denoted by F_i, which is used to supply the output buffer of the stage. We assume that raw parts are always available at the input of the system. Demands which cannot be immediately satisfied are backlogged in a queue denoted by D_N.

We consider the case of stages in series, and we assume that there is no blocking at the entry of each manufacturing system (i.e. the first queue of each manufacturing system has an infinite capacity, or the first station is an infinite server station) and at the exit

(i.e. parts are never blocked in the manufacturing process due to what happens outside the manufacturing process). Note that some cases for which these assumptions are not satisfied have been considered in (Bollon 2001) and in (Bollon *et al.* 2004).

We also assume that we have a mono-product system and that processes can produce batches of size Q_i at stage i. The size of batches for demands is Q_{N+1}. We assume that Q_i is a multiple of Q_{i+1}, and $Q_{N+1} = 1$, as usually assumed when dealing with batches (see (Axster and Rosling 1993) and (Liberopoulos and Dallery 2003) for example). This choice prevents the system from having useless remaining parts in a stage, and it also enables us to simplify the calculations.

In each manufacturing process, we use a push control mechanism, but the different stages are coordinated using a pull mechanism. With a pull control policy the production is pulled downstream by the demand. Whenever a demand is received, it is transmitted directly or indirectly to upstream stages in order to maintain a certain level in stock. The way this transmission of information is done depends on the control policy that is used.

When a part leaves a manufacturing process or a stock, or when a demand arrives, an information can be sent upstream the system to allow one or several parts to be processed. For example in a kanban policy when a part leaves a stock, a kanban label is sent to the upstream manufacturing process. This kanban allows a new part to enter the process. For a basestock policy, each demand is sent to all the manufacturing processes inputs to allow the process of a new part.

The processing times or the times between two demand arrivals can be either deterministic or stochastic, with general distribution. We only assume that the control mechanism is instantaneous.

3. Principles of our formulation of pull control policies

In this section, the unified framework initially proposed in (Bollon *et al.* 2004) for the description and the comparison of basestock, kanban, extended kanban and generalized kanban policies is described. Some elements about path algebra that are necessary to the derivation of the new algorithms presented in this paper are also given.

Let us first define some elements used to describe the dynamics of the system.

3.1. *Elements and notations for our unified formulation of policies*

We use a capital letter with an upright font to denote objects like manufacturing processes F_i or stores P_i. The scalar value for the number of parts present in an object is denoted by a small letter written with an italic font. For example, the number of parts in F_i, P_i or D_N are respectively f_i, p_i and d_N. If we need the value of these quantities at a given time t, we write $f_i(t)$, $p_i(t)$, etc.

We prove in (Bollon 2001), that the state of the system can be expressed with a state vector $\vec{X}$, introduced in (Veatch and Wein 1994). This vector $\vec{X}$, whose components are denoted by x_i, for $i = 1, \cdots, N$, is defined as follows

$$x_N = p_N - d_N \quad \text{and} \quad x_i = p_i + f_{i+1} \quad \text{for } i = 1, \cdots, N-1. \tag{1}$$

These components are illustrated in figure 1. A component x_i is equal to the sum of

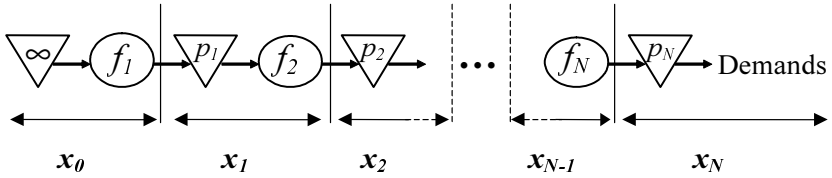


Figure 1. State variable for an N-stage line

parts present between two consecutive manufacturing outputs. When the quantity x_N is positive, it represents the number of available finished goods and when it is negative, it represents the number of unsatisfied demands that are backordered.

The vector $\vec{X}$ gives the size of the different inventory positions which are always positive except for the last one, which represents finished goods minus demands.

Each occurrence of an event (an outgoing part from a manufacturing process or a demand arrival) changes the vector $\vec{X}$. Conversely, if a vector change is known, then the event that has occurred is known too. In other words, all the events are detected if the vector $\vec{X}$ is known at every moment.

Let us now consider the values f_i. When they can be expressed as functions of $\vec{X}$, they are denoted by $f_i(\vec{X})$[1]. The components of the N-dimensional function $\vec{F}(\vec{X})$ are defined by $f_i(\vec{X})$ for i varying from 1 to N. The knowledge of this vector $\vec{F}(\vec{X})$ enables us to find the dynamics of the system: if $\vec{X}$ is known at every moment, events that occur are known, and the state of the system given by the values of p_i, f_i and d_N is known with equation (1).

Then, to describe the policy, we only have to search for $\vec{F}(\vec{X})$. For this, we use some tools described in the next section.

3.2. *Path algebra in dioids*

We present here some basic definitions of $(\min, +)$ algebra and its relation with graphs and shortest path search, which are tools that we use in our developments (see (Baccelli *et al.* 1992) for more details on this algebra).

3.2.1. Basic notions of $(\min, +)$ *algebra.*

We consider the set $\mathbb{R} \cup \{+\infty\}$ associated with the min operator (denoted by $\oplus$) and the usual addition (denoted by $\otimes$). This algebraic structure is called $\mathbb{R}_{\min}$ and is a dioid. The neutral element for $\oplus$ is $\varepsilon = +\infty$. ε is absorbing for $\otimes$. The neutral element for $\otimes$ is $e = 0$. In $\mathbb{R}_{\min}$ an inequality $e \succeq f$ is equivalent to $e \oplus f = e$, which means $e \leqslant f$, with usual notations.

Note that if we denote by $\oplus$ the *max* operator and by $\otimes$ the usual addition, then the set $(\mathbb{R} \cup \{-\infty\}$,$\oplus$, $\otimes)$ also has a dioid structure and is called (max,+) algebra.

The $(\min, +)$ algebra and the $(\max, +)$ algebra are often used to give a linear behavior representation of so called Timed Event Graphs (or TEG). These graphs are special cases of timed Petri Nets. They are composed of places (circles) and transitions (bars) and are such that each place has exactly one upstream and one downstream transition (see figure 2-a-). A transition is "fired" once each upstream place contains at least one available token (small black circle). Then one token is removed from each upstream place and one

[1]Note that f_i can be a function of the time (then we have a scalar value inside the brackets) or a function of the state (then we have a vector inside the brackets).

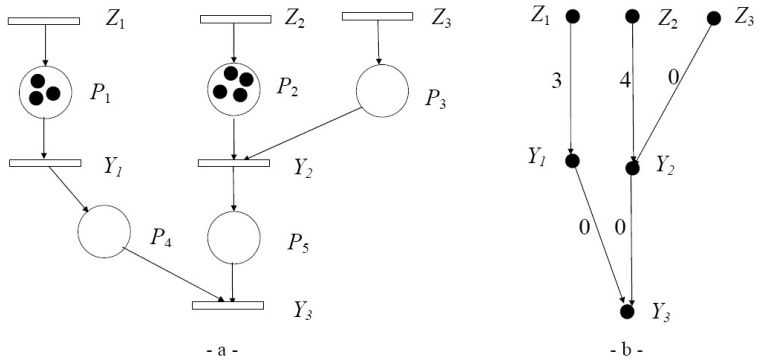


Figure 2. Example of a Timed Event Graph and its associated graph

token is added to each downstream place. A time delay can be associated with a place: the token becomes available to fire the downstream transition only when this delay has been consumed. In this paper, we are interested in a special case of a TEG, where all delays are zero, and transitions are fired at the earliest. Figure 2-a- shows a TEG with 5 places and 6 transitions, where all delays are zero. Transitions Z_i have no upstream places and are called input transitions.

3.2.2. Linear representation of Timed Event Graphs.

Two kinds of variables are used to describe the system state. They are associated with transitions. The first ones are counters: they count the number of tokens that have gone through a transition by a certain time. The other ones are daters, which give the moment when a transition is fired for the n^{th} time. At the beginning, all the daters and counters have the value zero. In this paper, we only deal with counters. For the TEG of figure 2-a-, the counters are denoted by $z_1(t)$, $z_2(t)$, $z_3(t)$, $y_1(t)$, $y_2(t)$ and $y_3(t)$. For example, $y_1(t)$ is the number of tokens that have gone through transition Y_1 at time t. Some relations linking these counters can be written to describe the dynamics of the system. We give below the inequalities obtained for the example of figure 2-a-.

$$\begin{aligned} y_1(t) &\leqslant z_1(t) + 3 \\ y_2(t) &\leqslant \min\left(z_2(t) + 4,\, z_3(t)\right) \\ y_3(t) &\leqslant \min\left(y_1(t), y_2(t)\right). \end{aligned} \tag{2}$$

We can then write relations (2) in $\mathbb{R}_{\min}$ as follows:

$$\begin{aligned} y_1(t) &\succcurlyeq 3 \otimes z_1(t) \\ y_2(t) &\succcurlyeq 4 \otimes z_2(t) \oplus z_3(t) \\ y_3(t) &\succcurlyeq y_1(t) \oplus y_2(t). \end{aligned} \tag{3}$$

Note that an equivalent dual system of inequalities can be given with the $(\max, +)$ algebra and the daters.

The use of $(\min, +)$ algebra and its notations enables us to obtain concise linear expressions for the formulations we are looking for. We will see now that the use of this algebra

and the associated results enables us also to facilitate the calculation, as explained in the following.

For the special case of TEG with all delays equal to zero, that we consider here, it is always possible to express every counter function $y_i(t)$ with input counters $z_i(t)$ and the initial number of tokens of each place using linear recurrences over the $(\min, +)$ algebra.

In other words, if we denote by $\vec{Z(t)} \in \mathbb{R}^m_{\min}$ the vector having the counters $z_i(t)$ of the m input transitions Z_i as its components, and by $\vec{Y(t)} \in \mathbb{R}^n_{\min}$ the vector having the counters of the n other transitions Y_i as its components, then $\vec{Y(t)}$ is the solution of the following system of inequalities (note that the symbol $\otimes$ has been omitted, as for the usual multiplication):

$$\vec{Y(t)} \succeq A\,\vec{Y(t)} \oplus B\,\vec{Z(t)}, \tag{4}$$

where A and B are matrices which belong respectively to $\mathbb{R}^{n\times n}_{\min}$ and $\mathbb{R}^{n\times m}_{\min}$.
For the example of figure 2-a- they are given by:

$$A = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ e & e & \varepsilon \end{bmatrix} \text{ and } B = \begin{bmatrix} 3 & \varepsilon & \varepsilon \\ \varepsilon & 4 & e \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix}.$$

3.2.3. Path algebra.

The smallest solution of system (4) can be given in an explicit way, using the "Kleene star" operator A^*:

$$\vec{Y(t)} = A^* B\,\vec{Z(t)} \qquad \text{with} \qquad A^* = \bigoplus_{k\geqslant 0} A^k, \tag{5}$$

where the product of two matrices A (belonging to $\mathbb{R}^{n\times n}_{\min}$) and B (belonging to $\mathbb{R}^{n\times m}_{\min}$) is defined as a matrix belonging to $\mathbb{R}^{n\times m}_{\min}$ denoted by AB (or A^2 if $B = A$), whose components $(AB)_{ij}$ are given by $\bigoplus_{k=1}^{k=n} A_{ik}B_{kj}$. A^0 is the identity matrix in $\mathbb{R}_{\min}$, where each component is equal to $e = 0$ on the diagonal and $\varepsilon = +\infty$ for the other elements.

This solution can be calculated using a shortest path search in a graph where vertices correspond to transitions and arcs correspond to places. Near a place there are two transitions, one before and one after, the vertices related to them are respectively the beginning and the ending of the arc associated with this place. The initial number of tokens present in the place will be the weight of the arc. Figure 2-b- gives the graph associated with the TEG of figure 2-a-.

Then, the value of A^*_{ij} gives the weight of the shortest directed path from Y_j to Y_i. Note that $A^*_{ij} = \varepsilon$ if there is no path from j to i. Knowing that a component B_{ij} gives the weight of the arc from Z_j to Y_i, then a component G_{ij} of matrix $G = A^*B$ represents the weight of the shortest directed path from Z_i to Y_j.

Then the solution (5) can be written equivalently:

$$y_i(t) = \bigoplus_{j=1}^{m} (G_{ji} \otimes z_j(t)) \quad \text{for } i = 0, \cdots, n\ . \tag{6}$$

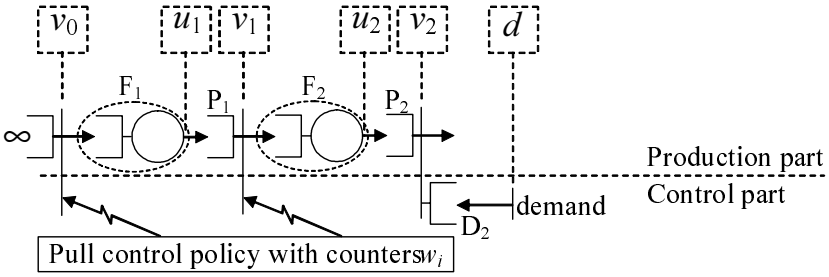


Figure 3. Main counters of a two-stage production line

Note that as a TEG never contains a circuit of negative weight, then matrix A^* is always a finite matrix and is given by:

$$A^* = \bigoplus_{k=0}^{k=n-1} A^k. \tag{7}$$

This link between $(\min, +)$ algebra and the graphs through the Kleene star, enables us to derive efficient algorithms to compute the canonical formulations, as we are going to see in the next sections.

The control mechanisms in a make-to-stock pull control policy can be expressed by an event graph. In this case $(\min, +)$ algebra will be useful to describe the policies. Note that an equivalent representation of an event graph can be obtained using a queueing network. It is this queueing network representation that we are using in the remainder of the paper.

3.3. *Structure of pull control systems and required counters*

A production line using a pull control policy can be divided into two parts (see figure 3):

- The first part is composed of stocks P_i and manufacturing processes F_i where the production flow takes place. Each stock is connected to a synchronization station on which the second part of the system acts.
- The second part is the mechanism of the pull policy. It controls the production requirement flow of information moving upstream the line.

In the production part of the system we use a counter at the end of each manufacturing process F_i (represented by an oval in figure 3) and each buffer P_i (represented by a queue, linked to a synchronization station in figure 3). The first counter, denoted by u_i, detects the outgoing of products from the manufacturing process and the other, denoted by v_i, counts the release of available parts from stocks. If a synchronization station coincides with a counter u_i or v_i, then we denote it by U_i or V_i respectively. In the control part of the system, an input counter d is used to detect the arrival of demands. The pull control policy may require additional synchronization stations W_i, which we associate with counters denoted by w_i.

For an N-stage line, an arc connects the arrival of a demand, represented by a transition called D, to a queue D_N at the synchronization station V_N. Depending on the policy, some other arcs move from a transition D, V_i, U_i or W_i to an upstream transition V_j or W_j.

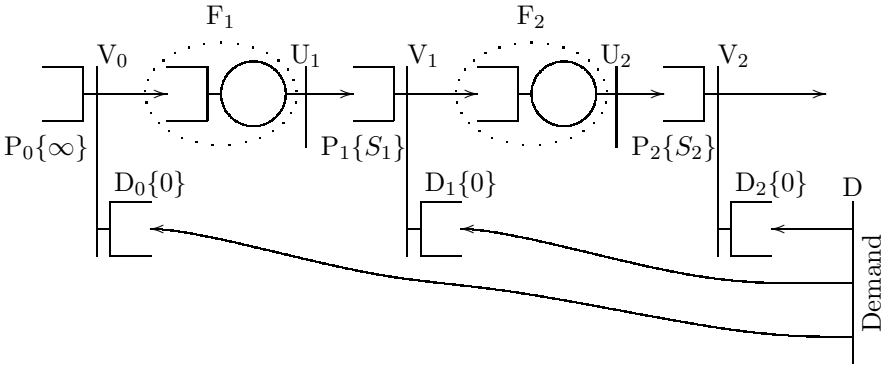


Figure 4. Counters for a two-stage basestock system

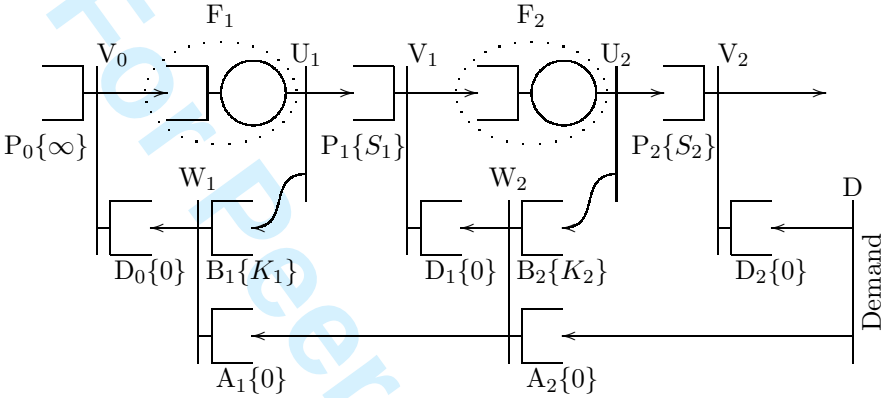


Figure 5. Counters for a two-stage generalized kanban system

Figures 4 and 5 illustrate the notations and counters for a two stage basestock system and a two stage generalized kanban system. The queues are labelled according to their content (P_i, D_i,...) and their initial value is indicated inside braces. Note that for both systems, each arriving demand generates a production order in order to maintain buffers P_i at their nominal levels denoted by S_i. In the base stock system, this information is immediately transmitted to all the stages, (via the synchronization stations V_i), whereas for the generalized kanban, these production orders are not transmitted instantaneously upstream since they must be associated with a free kanban (synchronization with queues B_i). The production orders, waiting for free kanbans (in queues B_i) to be transmitted, are put in queues A_i. There are thus additional synchronization stations called W_i (and associated with counters w_i). The K_i kanbans of a stage i circulate around the input and the output of the processes F_i. They move from the input to the output of the process attached to a part. To move from the output to the input of the process they must be associated with a demand from downstream. Initially, when $t = 0$, we assume that $f_i(0) = 0$, $d_i(0) = 0$, $p_i(0) = S_i$ and, for the generalized kanban, $a_i(0) = 0$, $b_i(0) = K_i$.

To calculate the $\vec{F}(\vec{X})$ function, we first use the previously defined counters to obtain the function of the time $\vec{F}(t)$ whose components are:

$$f_i(t) = f_i(0) + v_{i-1}(t) - u_i(t) \quad \text{for } i = 1, \cdots, N \ . \tag{8}$$

Considering that the control part of the system is an event graph and that the $u_i(t)$ and $d(t)$ are inputs of this event graph, then the $v_{i-1}(t)$ can be expressed in terms of

$u_i(t)$ and $d(t)$ using a $(\min, +)$ algebra. Moreover counters $u_i(t)$ and $d(t)$ can be linked to variables x_i through equations (9):

$$\begin{aligned} x_i(t) - x_i(0) &= u_i(t) - u_{i+1}(t) \text{ for } i = 1, \cdots, N-1 \text{ and} \\ x_N(t) - x_N(0) &= u_N(t) - d(t). \end{aligned} \tag{9}$$

They can then be removed and replaced by an expression using $\vec{X}(t)$ in equation (8). If all the counters are removed and replaced by the $\vec{X}(t)$ components, we can obtain the function $\vec{F}(\vec{X})$ from $\vec{F}(t)$ by substituting $\vec{X}$ for $\vec{X}(t)$.

3.4. *A canonical formulation*

In (Bollon *et al.* 2004), we calculated the functions $\vec{F}(\vec{X})$ for four pull control policies: basestock, kanban, extended kanban and generalized kanban. For these calculations we used the $(\min, +)$ algebra notations that facilitated calculations and enabled us to obtain concise expressions.

For these policies, the function $f_i(\vec{X})$ always has the same structure, that we call the canonical formulation. This formulation is given in equation (10), where the fractions stand for the usual minus:

$$\begin{aligned} f_1\left(\vec{X}\right) &= \bigoplus_{j=1}^{N+1} \left(C_{(1,j)} \Big/ \bigotimes_{k=1}^{j-1} x_k \right) \quad \text{and} \\ f_i\left(\vec{X}\right) &= \bigoplus_{j=i}^{N+1} \left(C_{(i,j)} \Big/ \bigotimes_{k=i}^{j-1} x_k \right) \oplus x_{i-1} \text{ for } i = 2, \cdots, N. \end{aligned} \tag{10}$$

The parameters $C_{(i,j)}$ of this formulation define the policy and are calculated from the control parameters of each policy, namely, S_k, the maximum level of finished products of each stage k, and K_k, the number of kanbans in each stage k. For example, for a basestock control policy, the parameters $C_{(i,j)}$, for $1 \leqslant i \leqslant j \leqslant N$, are:

$$C_{(i,N+1)} = \bigotimes_{k=i}^{N} S_k \quad \text{and} \quad C_{(i,j)} = \varepsilon \tag{11}$$

and for a generalized kanban control system the parameters $C_{(i,j)}$, for $1 \leqslant i \leqslant j \leqslant N$, are:

$$C_{(i,N+1)} = \bigotimes_{k=i}^{N} S_k \quad \text{and} \quad C_{(i,j)} = K_j \otimes \bigotimes_{k=i}^{j-1} S_k. \tag{12}$$

Note that for each parameter $C_{(i,j)}$, a control cell, like those introduced in (Bonvik 1996), including processes and stocks from F_i to F_j can be defined. In that control cell, the number of parts should never exceed $C_{(i,j)}$.

In the next section, we show that a canonical formulation can be found for many other pull policies and we derive an algorithm that enables us to compute this formulation

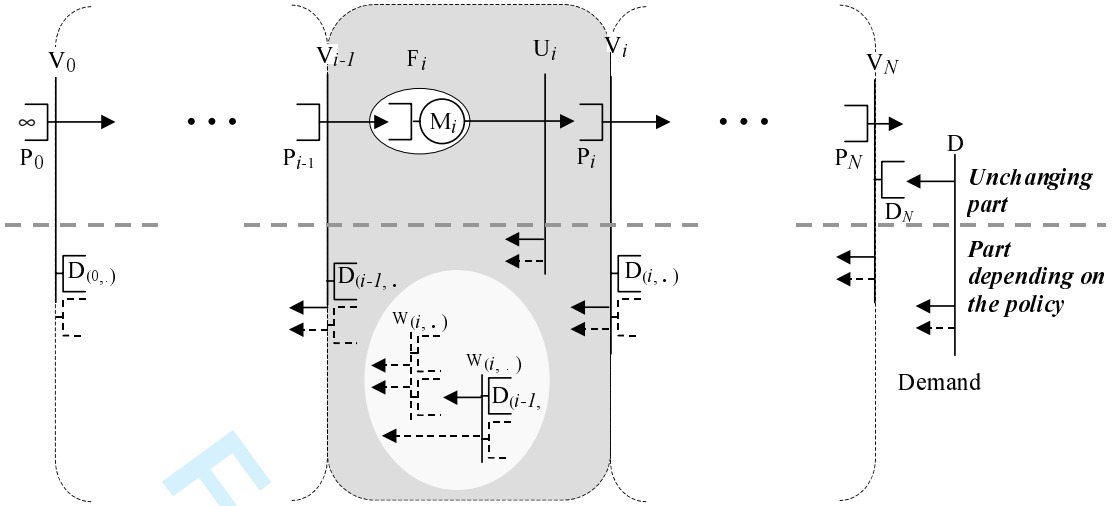


Figure 6. Description of a class of policies having a canonical formulation

efficiently for all the policies. Then, in section 5, we extend this work to systems producing batches.

4. Canonical formulation for more general pull control policies without batch production

4.1. *Existence for a class of pull policies*

In this part we are going to show that it is possible to describe a class of pull policies with our canonical formulation. This class is defined in proposition 4.1 and is illustrated in figure 6.

In figure 6, a generic stage i is represented in grey. Most of the queues and transitions are not necessarily present. Entities present in the upper part of figure 6 (called "unchanging part") are required. Those present in the lower part of the figure (called "part depending on the policy") can appear one or several times, or can be dismissed. So manufacturing processes F_i, stocks P_i, queue D_N, synchronization stations V_{i-1}, U_i and D, arcs going downstream and the arc going from D to D_N are all required. The arcs that are going upstream can reach any of the upstream queues that are oriented in the same direction. The synchronization station U_i which follows F_i enables us to locate the counter u_i and it often contains only one arc and has no utility for the policy itself (as it was the case for the basestock policy in figure 4) .

Note that this class of policies, described in proposition 4.1 below, includes basestock (figure 4), kanban, generalized kanban (figure 5), extended kanban and many other hybrid policies, like those presented in (Bonvik 1996) and in (Liberopoulos and Dallery 2000). An example of such a hybrid policy (CONWIP/Kanban) is given in figure 7.

Proposition 4.1: *An N-stage queueing production system has a canonical formulation* $F_i(\vec{X})$ *for* $i = 1, \cdots, N$ *given by equation* (10)*, if it is defined as follows:*

- *In its production part the queueing system is composed of:*
  - *Manufacturing processes* F_i *which are composed of a queue followed by a station* M_i*.*
  - *Synchronization stations* U_i *(associated with counters* $u_i(t)$*) having a unique input coming from* M_i *and an output going to queue* P_i*.*

- *Stocks with queues* P_i. *A queue* P_0 *containing raw parts, and thus never empty, is added.*
- *Synchronization stations* V_i, *for* $i = 0, \cdots, N$ *(associated with counters* $v_i(t)$*) having an input coming from* P_i *and an output going to* F_{i+1}. *Note that* V_N *has an output going outside the system.*

- *In the control part depending on the policy, the system can be composed of:*
  - *Queues* $\mathrm{D}_{(i,j)}$ *feeding* V_i *or* $\mathrm{W}_{(i+1,k)}$, *for* $i = 0, \cdots, N-1$ *and* $j, k \in \mathbb{N}$.
  - *A queue* D_N *feeding* V_N *and fed by* D.
  - *Synchronization stations* V_i, U_i *or* $\mathrm{W}_{(i,j)}$ *(respectively associated with counters* $v_i(t)$, $u_i(t)$ *and* $w_{(i,j)}(t)$*) with outputs going to some queues* $\mathrm{D}_{(k-1,l)}$ *feeding* V_{k-1} *or* $\mathrm{W}_{(k,m)}$, *where* $1 \leqslant k \leqslant i \leqslant N$ *and* $l, m \in \mathbb{N}$. *In a same stage we assume that synchronization stations* $\mathrm{W}_{(i,j)}$ *are ordered so that* $\mathrm{W}_{(i,j)}$*can not feed* $\mathrm{W}_{(i,m)}$ *for* $j \leqslant m$.

Proof: *of proposition 4.1*

The values for $f_i(t)$ are given by equation (8). The difficulty relies in the way to express counters v_i as functions of counters u_i. The queueing network for the control mechanisms of the make-to-stock pull control policy, illustrated in figure 6, can be expressed by an event graph with no delay. Synchronization stations correspond to transitions and queues correspond to places. So a system of inequalities as in (4) can be set when describing the dynamics of the event graph. For this system, input transitions (Z_i in (4)) are U_i and D. Other transitions (Y_i in (4)) are V_i and $\mathrm{W}_{(i,j)}$. Matrices A and B of relation (4) can be obtained line by line by describing each transition by equations (as we did for the TEG of figure 2).

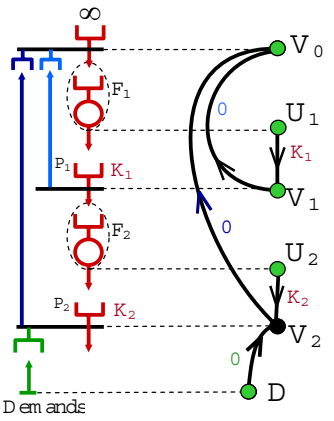
The solution (5) of this system allows to express counters $v_i(t)$ and $w_{(i,j)}(t)$ as functions of counters $u_k(t)$ and $d(t)$. This solution can be calculated using a shortest path search, as shown in section 3.2.3. For this, we define a graph G (see figure 7 for an example) where vertices correspond to synchronization stations and where arcs between vertices correspond to arcs between synchronization stations (note that stations F_i are not represented in these graphs). The initial number of tokens in a queue at the end of an arc gives the weight of this arc in the graph G. Counters v_i and w_i can be expressed as functions of counters u_j, for $j = 1, \cdots, N$, and d. Note that for finding the f_i functions, we only need to express counters v_i as functions of counters u_j and d (see equation (8)). Let us denote by $g(i,j)$ the weight of the shortest path in G from U_i to V_j and by $g(i)$ the weight of the shortest path in G from D to V_i. Then the solution (5) of the system of inequalities (4) can be written equivalently as in equation (6):

$$v_i(t) = \bigoplus_{j=1}^{N} \left(g(j,i) \otimes u_j(t)\right) \oplus \left(g(i) \otimes d(t)\right) \quad \text{for } i = 0, \cdots, N \ . \tag{13}$$

For a make-to-stock pull control policy following the definition of proposition 4.1 we have (remember that $p_j(0)$ denotes the content of queue P_j at time 0) :

$$g(j,j) = p_j(0) \quad \text{and} \quad g(j,i) = \varepsilon \qquad \text{for } 1 \leqslant j < i \leqslant N \ . \tag{14}$$

So, after a simplification, the index j in equation (13) will not run from 1 to N but from 1 to i. From equations (8), (9), (13) and (14) we obtain the canonical formulation given



$$C_{(1,1)}=K_1$$
$$C_{(1,2)}=K_1 \otimes K_2$$
$$C_{(1,3)}=K_1 \otimes K_2 \otimes e = K_1 \otimes K_2$$
$$f_1 = C_{(1,1)} \oplus C_{(1,2)}/x_1 \oplus C_{(1,3)}/x_1/x_2$$

$$C_{(2,2)}= \varepsilon$$
$$C_{(2,3)}=K_2 \otimes \varepsilon = \varepsilon$$
$$f_2 = x_1 \oplus C_{(2,2)} \oplus C_{(2,3)}/x_2$$
$$= x_1$$

Figure 7. Algorithm 4.2 for a CONWIP/Kanban policy

in equation (10). Parameters $C_{(i,j)}$ are then given by:

$$C_{(i,j)} = \left(\bigotimes_{k=i}^{j-1} p_k(0) \right) \otimes g(j, i-1) \quad \text{and}$$
$$C_{(i,N+1)} = \left(\bigotimes_{k=i}^{N} p_k(0) \right) \otimes g(i-1) \quad \text{for } 1 \leqslant i \leqslant j \leqslant N. \tag{15}$$

□

The proposition is thus proved and, through the use of equation (15), this proof enables us to derive the algorithm 4.2 presented in the next section to compute the canonical formulation.

4.2. *An algorithm to compute the canonical formulation*

The algorithm 4.2 below is able to calculate the canonical formulation for any policy consistent with proposition 4.1. It uses the links that exist between $(min, +)$ algebra and the graphs, as shown in the proof of proposition 4.1. The core of this algorithm is a shortest path search[1]. In figure 7 the algorithm is running for the CONWIP/Kanban hybrid policy (see (Bonvik 1996) for more details on this policy). From left to right, one can see the queueing network of this policy, the temporary graph G used for calculation, and the computations for parameters $C_{(i,j)}$ of the canonical formulation.

Algorithm 4.2 We assume that $f_i(0) = 0$ for i varying from 1 to N.

// Construction of a temporary graph G used for calculations
Add vertices to the empty graph G, corresponding to each V_i, U_i, D and $\mathrm{W}_{(i,j)}$; name them as the associated synchronization stations.

[1]Note that as all the weights on the arcs have positive values, we can use Dijkstra algorithm to compute the weight of the shortest path of the graph

Add an arc from a vertex to another every time an arc exists between the corresponding synchronization stations. No arc should be added from V_{i-1} to U_i. The weight is given by the initial number of tokens present in the queue at the end of the arc.

If (considering that there is an arc of weight $f_i(0)$ between V_{i-1} and U_i a zero length cycle exists) **then**
Print " System is not alive"; **exit**
End if

// f_i calculation :
For i from 1 to N **do**
For j from i to N **do**
$C_{(i,j)} = \left(\bigotimes_{k=i}^{j-1} p_k(0) \right) \otimes$ weight of the shortest oriented path from U_j to V_{i-1} in G
End for j
$C_{(i,N+1)} = \left(\bigotimes_{k=i}^{N} p_k(0) \right) \otimes$ weight of the shortest oriented path from D to V_{i-1} in G
If $i = 1$ **then**
Print $f_1\left(\vec{X}\right) = \bigoplus_{j=1}^{N+1} \left(C_{(1,j)} \Big/ \bigotimes_{k=1}^{j-1} x_k \right)$
Else
Print $f_i\left(\vec{X}\right) = \bigoplus_{j=i}^{N+1} \left(C_{(i,j)} \Big/ \bigotimes_{k=i}^{j-1} x_k \right) \oplus x_{i-1}$
End if
End for i

Note that when a zero length cycle is found in the graph obtained from G by adding arcs of weight $f_i(0) = 0$ from each V_{i-1} to U_i, there is no solution because the system is not alive. For example a kanban policy with a parameter K_i equal to zero is not alive.

In next section, we show how these computations can be extended to systems producing batches.

5. Systems producing batches

5.1. *Existence of a canonical formulation for a class of pull control policies producing batches*

Let us consider the category of pull control policies described by proposition 4.1. We now assume that processes are producing batches of size Q_i at stage i. The batch size of demands is Q_{N+1}. As in (Axster and Rosling 1993), we assume that Q_i is a multiple of Q_{i+1}, and $Q_{N+1} = 1$. These choices prevent the system from having useless remaining parts in a stage, and they also allow us to simplify the results.

First let us consider the case when processes are composed of an infinite input queue followed by a server processing batches of size Q_i. Batches of size one are then allowed to enter the infinite input queue, but batches of size Q_i are outgoing from the process. Consequently nothing changes for the computation of the canonical formulation (10), and the number of parts in process for stage i is given by $f_i(\vec{X})$.

If we now consider that, at each stage i, inputs and outputs are batches of size Q_i for the process composed of an infinite queue followed by a station, then processes may create some blocking. The synchronization station preceding the process must also deliver batches of size Q_i for parts and control information. To compute the functions $f_i(\vec{X})$, algorithm 4.2 can no longer be used. However, for a class of pull policy with batches, it is possible to use a similar algorithm with a new notation[1], as shown in algorithm 2. Note that this class of policies with batches, described in proposition 5.1 below, includes the installation and echelon kanban policies illustrated in figures 8 and 9, the installation and the echelon stock (Q,r)-policies described in (Axster and Rosling 1993), and the hybrid policies described in (Liberopoulos and Dallery 2003).

Proposition 5.1: *The policy of a queueing system producing batches can be defined by the functions*

$$f_1\left(\vec{X}\right) = \left\lfloor \bigoplus_{j=0}^{N} \left(C_{(1,j)} \Big/ \bigotimes_{k=1}^{j} \lfloor x_k \rfloor_{Q_{k+1}} \right) \right\rfloor_{Q_1} \quad \text{and}$$
$$f_i\left(\vec{X}\right) = \left\lfloor \bigoplus_{j=i-1}^{N} \left(C_{(i,j)} \Big/ \bigotimes_{k=i}^{j} \lfloor x_k \rfloor_{Q_{k+1}} \right) \oplus \lfloor x_{i-1} \rfloor_{Q_i} \right\rfloor_{Q_i} \quad \text{for } i = 2, \cdots, N. \tag{16}$$

if it is defined as follows:

- *The queueing system in its production part is composed of:*
  - *Manufacturing processes* F_i *including an infinite queue followed by a station* M_i *producing batches of size* Q_i*. Inputs and outputs of the process are batches of size* Q_i*.*
  - *Synchronization stations* U_i *(associated with the batch counter* $u_i(t)$*) having a unique input for batches of size* Q_i *coming from* M_i *and an output going to the queue* P_i*. Each output of* U_i *delivers batches of size* Q_i*.*
  - *Stocks with queues* P_i*. The raw parts queue* P_0 *is never empty.*
  - *Synchronization stations* V_i *(associated with the batch counter* $v_i(t)$*) having an input coming from* P_i *and an output going to* F_{i+1} *or out of the system for* $i = N$*. Every output from* V_i *are batches of size* Q_{i+1}*. When batches are outgoing from* V_i*,* Q_{i+1} *parts are taken in each queue linked with it.*
- *In the control part, the system is composed of:*
  - *Queues* $\mathrm{D}_{(i,j)}$ *feeding* V_i *or* $\mathrm{W}_{(i+1,k)}$*, for* $i = 0, \cdots, N-1$ *and* $j, k \in \mathbb{N}$*.*
  - *A queue* D_N *feeding* V_N *and fed by* D*.*
  - *Synchronization stations* V_i*,* U_i *or* $\mathrm{W}_{(i,j)}$ *(respectively associated with counters* $v_i(t)$*,* $u_i(t)$ *and* $w_{(i,j)}(t)$*) with outputs going to some queues* $\mathrm{D}_{(k-1,l)}$ *feeding* V_{k-1} *or* $\mathrm{W}_{(k,m)}$*, where* $1 \leqslant k \leqslant i \leqslant N$ *and* $l, m \in \mathbb{N}$*. In a same stage we assume that synchronization stations* $\mathrm{W}_{(i,j)}$ *are ordered so that* $\mathrm{W}_{(i,j)}$*can not feed* $\mathrm{W}_{(i,m)}$ *for* $j \leqslant m$*. When batches are outgoing from* $\mathrm{W}_{(i,j)}$*,* Q_i *parts are taken in each queue linked with it.*

Proposition 5.1 is similar to proposition 4.1 except that on each stage i, synchronization stations are making batches of size Q_{i+1}, and that we obtain a modified canonical

[1] we denote by $\lfloor \alpha \rfloor$ the highest integer smaller than or equal to α and by $\lfloor \alpha \rfloor_\beta = \beta \left\lfloor \frac{\alpha}{\beta} \right\rfloor$ the highest multiple of β smaller or equal to α

formulation.

To prove proposition 5.1, it is enough to prove algorithm 5.2 below, which builds the functions $f_i\left(\vec{X}\right)$ of equations (16).

5.2. *An algorithm to compute the canonical formulation for batch production systems*

Algorithm 5.2 This algorithm is used for the queueing production system defined in proposition 5.1 – notations are identical. We assume that $f_i(0) = 0$ for i varying from 1 to N.

// Construction of a temporary graph G used for calculation
Add vertices to the empty graph G, corresponding to each V_i, U_i, D and $\mathrm{W}_{(i,j)}$; name them as the associated synchronization stations.

Add an arc from a vertex U_k, V_{k-1}, $\mathrm{W}_{(k,l)}$ or D to a vertex V_{i-1} or $\mathrm{W}_{(i,m)}$ every time an arc exists between the corresponding synchronization stations. The weight on the arc is given by $\left\lfloor d_{(i-1,j)}\right\rfloor_{Q_k}$ (or $\left\lfloor d_{(i-1,j)}\right\rfloor_{Q_N+1}$ if the arc is coming from D) where $d_{(i-1,j)}(0)$ is the initial number of tokens present in the queue $\mathrm{D}_{(i-1,j)}$ at the end of the arc.

Add an arc from the vertex U_k to the vertex V_k, for k varying from 1 to N. The weight on the arc is given by $\lfloor p_k(0)\rfloor_{Q_{k+1}}$.

// f_i calculations :
For i from 1 to N **do**
For j from i to N **do**

$$C_{(i,j)} = \left(\bigotimes_{k=i}^{j-1} \lfloor p_k(0)\rfloor_{Q_{k+1}}\right) \otimes \text{weight of the shortest oriented path from } \mathrm{U}_j \text{ to } \mathrm{V}_{i-1} \text{ in } G$$

If $\left(\left\lfloor C_{(i,j)}\right\rfloor_{Q_i} = e\right)$ **then print** "System is not alive"

End for j

$$C_{(i,N+1)} = \frac{\left(\bigotimes_{k=i}^{N} \lfloor p_k(0)\rfloor_{Q_{k+1}}\right)}{d_N(0)} \otimes \text{weight of the shortest oriented path from D to } \mathrm{V}_{i-1} \text{ in } G$$

If $i = 1$ **then**

$$\textbf{Print} \quad f_1\left(\vec{X}\right) = \left\lfloor \bigoplus_{j=1}^{N+1} \left(C_{(1,j)} \Big/ \bigotimes_{k=1}^{j-1} \lfloor x_k\rfloor_{Q_{k+1}} \right) \right\rfloor_{Q_1}$$

Else

$$\textbf{Print} \quad f_i\left(\vec{X}\right) = \left\lfloor \bigoplus_{j=i}^{N+1} \left(C_{(i,j)} \Big/ \bigotimes_{k=i}^{j-1} \lfloor x_k\rfloor_{Q_{k+1}} \right) \oplus \lfloor x_{i-1}\rfloor_{Q_i} \right\rfloor_{Q_i}$$

End if
End for i

This algorithm is similar to algorithm 4.2. The number of customers in queues is rounded off to a certain multiple, so some customers may not always be useful. We give a proof for this algorithm 5.2 in appendix A.

Note that there is a deadlock in the system only if a loop, including a manufacturing process F_i can be found and the number of customers in this loop is not sufficient to enable the manufacturing process to work.

No cycle can be found in the graph G because vertices can be ordered in such a way that an arc is always going from a vertex to another one placed further away.

However, if $C_{(i,j)}$, for $j \neq N+1$, is not equal to ε (the infinite value), then there is a loop in the queueing network going upstream from U_j to V_{i-1} in the control part and downstream from V_{i-1} to U_j in the production part. If $\left\lfloor C_{(i,j)} \right\rfloor_{Q_i}$ is equal to e (the value zero) then for any state vector $\vec{X}$, $f_i(\vec{X})$ is equal to zero, which means that manufacturing F_i never works and consequently the system is not alive.

5.3. *Application of the algorithm for the formulation of some pull policies*

We are now going to apply proposition 5.1 to find the values of parameters $C_{(i,j)}$ for some pull policies encountered in the literature. Note that in algorithm 5.2 above, only the operator $\otimes$ (standing for the usual addition) appears in the expression giving $C_{(i,j)}$. Thus, in the remainder of the section, we will use the usual notations for the addition and the multiplication. The values of parameters $C_{(i,j)}$ are then given by:

$$\begin{aligned} C_{(i,j)} &= \left(\sum_{k=i}^{j-1} \lfloor p_k(0) \rfloor_{Q_{k+1}} \right) + g(j, i-1) \quad \text{and} \\ C_{(i,N+1)} &= \left(\sum_{k=i}^{N} \lfloor p_k(0) \rfloor_{Q_{k+1}} \right) - d_N(0) + g(i-1) \\ &\text{for } 1 \leqslant i \leqslant j \leqslant N, \end{aligned} \tag{17}$$

where $g(j, i-1)$ is the weight of the shortest path in G from U_j to V_{i-1} and $g(i-1)$ is the weight of the shortest path in G from D to V_{i-1}.

5.3.1. Description of the installation and echelon kanban policies

These installation and echelon kanban policies, derived from the kanban and introduced in (Liberopoulos and Dallery 2003), are illustrated by the queueing networks of figures 8 and 9 .

In such policies, there is a finite number of kanban cards, associated to each stage, which is an integer multiple of the stage batch size Q_i and will be denoted by $K_i Q_i$. A kanban may be either free or attached onto a part. When Q_i free i-stage kanbans have accumulated at stage i (in queue D_{i-1}), an order of size Q_i is placed at stage i. If Q_i parts are available in the output buffer P_{i-1} of stage $i-1$, the free kanbans are attached onto these parts and are sent to the manufacturing process F_i of stage i. The kanbans remain attached to the parts until they reach a certain *final output buffer*. When a part leaves this final output buffer for a given stage i, the kanban that was attached to it is detached and becomes a free kanban.

The difference between installation kanban and echelon kanban lies in the definition of the final output buffer, i.e. the point after which kanbans are detached from parts, which leads to a local transfer of information for the installation kanban policy and a

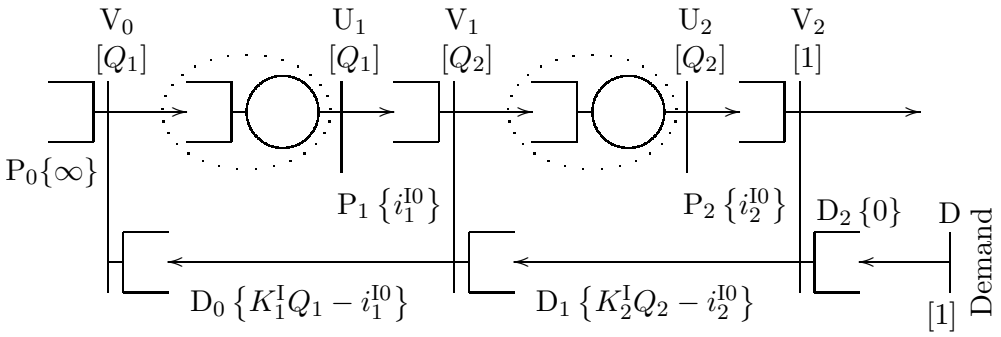


Figure 8. Queueing network model of a two-stage production system controlled with an installation kanban policy

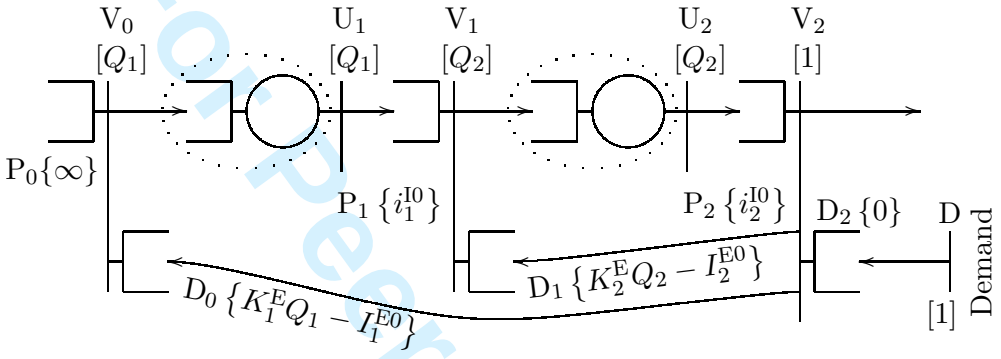


Figure 9. Queueing network model of a two-stage production system controlled with an echelon kanban policy

global transfer of information for the echelon kanban policy. In an installation kanban policy, the final output buffer at a stage i is the output buffer P_i of stage i. In an echelon kanban policy, the *final* output buffer at a stage i is the output buffer P_N of last stage N. The kanbans used in installation kanban and echelon kanban policies are respectively referred to as installation and echelon kanban.

Note that in an installation kanban policy, every part in the manufacturing process F_i or in the output buffer P_i of a given stage i has one stage-i installation kanban attached on it. In an echelon kanban policy, every part in the manufacturing process F_i or in the output buffer P_i of a given stage i has i echelon kanban attached on it, one from each of stages 1 to i. Thus, in an echelon kanban policy, when a finished part is consumed by a customer, N echelon kanbans are detached from the part and become free.

5.3.2. Calculation of $f_i(\vec{X})$ for the installation kanban policy

In order to calculate $f_i(\vec{X})$, it is sufficient to calculate the value of parameters $C_{(i,j)}$, and then use relations (16) to obtain $f_i(\vec{X})$. Consider the graph G obtained for the installation kanban, with algorithm 5.2. There is a path between each U_j and V_i for $i < j$, with the weight $\lfloor p_j(0) \rfloor_{Q_{j+1}} + \sum_{k=i}^{j} \lfloor d_{k-1}(0) \rfloor_{Q_{k+1}}$. Thus we obtain:

$$\begin{aligned} C_{(i,j)} &= \sum_{k=i}^{j} \left(\lfloor p_k(0) \rfloor_{Q_{k+1}} + \lfloor d_{k-1}(0) \rfloor_{Q_{k+1}} \right) \\ &= \sum_{k=i}^{j} \left(\lfloor i_k^{\mathrm{I0}} \rfloor_{Q_{k+1}} + \lfloor K_k^{\mathrm{I}} Q_k - i_k^{\mathrm{I0}} \rfloor_{Q_{k+1}} \right) \qquad \text{and} \\ C_{(i,N+1)} &= C_{(i,N)} \qquad\qquad \text{for } 1 \leqslant i \leqslant j \leqslant N. \end{aligned} \tag{18}$$

Parameters $C_{(i,j)}$ are using rounded values of queues level in P_k and D_k. It means that all about the control for this policy can be expressed with rounded values of these queues. So without loss of generality we assume that the initial installation stock i_k^{I0} is an integer multiple of Q_{k+1} for $k = 1, \cdots, N$ and parameters in (18) become:

$$\begin{aligned} C_{(i,j)} &= \sum_{k=i}^{j} (K_k^{\mathrm{I}} Q_k) \qquad \text{and} \\ C_{(i,N+1)} &= C_{(i,N)} \qquad \text{for } 1 \leqslant i \leqslant j \leqslant N, \end{aligned} \tag{19}$$

which shows that the behavior of the installation kanban policy does not depend on the initial installation stock i_i^{I0} but only on the batch sizes Q_i and the integers K_i for each stage i. Note that we obtain the same values for the parameters $C_{(i,j)}$ than for the kanban policy presented in (Bollon *et al.* 2004), if we consider that the number of kanbans in stage k is $K_k^{\mathrm{I}} Q_k$.

5.3.3. Calculation of $f_i(\vec{X})$ for the echelon kanban policies

Consider the graph G obtained for the echelon kanban, with algorithm 5.2. No path can be found between U_j and V_i for $i < j \leqslant N-1$. However, one path of weight $p_N(0) + d_{i-1}(0)$ is present between U_N and each transition V_{i-1}, and a path of weight $d_{i-1}(0)$ is present between d and each transition V_{i-1}. We thus obtain:

$$\begin{aligned} C_{(i,j)} &= +\infty \\ C_{(i,N+1)} = C_{(i,N)} &= \sum_{k=i}^{N} \lfloor p_k(0) \rfloor_{Q_{k+1}} + d_{i-1}(0) \\ &= \sum_{k=i}^{N} \lfloor i_k^{\mathrm{I0}} \rfloor_{Q_{k+1}} + K_i^{\mathrm{E}} Q_i - I_i^{\mathrm{E0}} \\ \text{for } i = 1, \cdots, N &\text{ and } j = i, \cdots, N-1 \ . \end{aligned} \tag{20}$$

Again, we assume that the initial installation stock i_k^{I0} is a multiple of Q_{k+1} for $k = 1, \cdots, N$, and the parameters in (20) become:

$$\begin{aligned} C_{(i,j)} &= +\infty \\ C_{(i,N+1)} = C_{(i,N)} &= K_i^{\mathrm{E}} Q_i \quad \text{for } i = 1, \cdots, N \text{ and } j = i, \cdots, N-1 \ . \end{aligned} \tag{21}$$

6. Possible uses of our formulation

The formulation presented in the previous sections can be used for a qualitative or a quantitative comparison of pull policies.

6.1. *Qualitative comparison between pull policies*

6.1.1. Identification of existing equivalences between some policies

We consider that two systems, with the same manufacturing process, have an identical behavior if they have the same dynamics, defined by vector $\vec{F}(\vec{X})$. When a manufacturing process contains only one server, it can be either on or off. A manufacturing process F_i is off if $f_i = 0$ and is on if $f_i \geqslant 1$. So, two policies have a same control if and only if all the functions $\min(1, f_i(\vec{X}))$ for $i = 1, \ldots, N$ are equal for any admissible state $\vec{X}$, for both policies.

For more general manufacturing processes, two policies have an identical behavior if and only if all the $f_i(\vec{X})$ functions are equal. In order to compare functions $f_i(\vec{X})$, we must examine the arguments of the min functions of the canonical formulation (10). These arguments are equal if parameters $C_{i,j}$ of the canonical formulation are equal for both policies.

If all the functions $f_i(\vec{X})$ obtained for a given policy are lower than the ones obtained for a second policy, then, the production capacity of the first policy is always lower than the one for the second policy. For example, if an extended kanban system and a generalized kanban system have identical parameters, then for any given state vector $\vec{X}$, the number of parts in each process of the extended kanban system F_i is always greater than or equal to the number of parts in each process of the generalized kanban system (compare functions $f_i(\vec{X})$ for i varying from 1 to N). So, with the same parameters, the production capacity for an N-stage extended kanban system is higher than the production capacity for an N-stage generalized kanban system as pointed out in (Dallery and Liberopoulos 2000).

Thus, by comparing the values of the $f_i(\vec{X})$ functions, we can find under which conditions a policy has the same behavior as another one. In (Bollon *et al.* 2004), we compared extended kanban and generalized kanban policies this way, using a polyhedral comparison, and we found all the possible parameter values for which the extended kanban and the generalized kanban have the same dynamics.

As an illustration, let us consider now again the installation kanban and echelon kanban policies presented in section 5.3.1.

Remember that for each parameter $C_{(i,j)} \neq +\infty$, a control cell can be defined, in which the number of customers is limited by $C_{(i,j)}$. The echelon and installation stock kanban policies are controlling the inventory positions $\sum_{k=i}^{N} \lfloor x_k \rfloor_{Q_{k+1}}$ which have to reach the levels $C_{(i,N+1)}$ without exceeding them for $i = 1, \cdots, N$.

For the echelon kanban policy a control is done on the inventory position $\sum_{k=i}^{N} \lfloor x_k \rfloor_{Q_{k+1}}$

with maximum level $C_{(i,N+1)}$ but also on the inventory position $\sum_{k=i}^{N-1} \lfloor x_k \rfloor_{Q_{k+1}}$ with maximum levels $C_{(i,N)}$ for $i = 1, 2, \cdots, N$.

For the installation kanban policy a control is done on the inventory positions $\sum_{k=i}^{j} \lfloor x_k \rfloor_{Q_{k+1}}$ with maximum levels $C_{(i,j+1)}$ for $i = 1, 2, \cdots, N$ and for $j = 1, 2, \cdots, N$. Parameters $C_{(i,i)}$ for $i = 1, 2, \cdots, N$ control the maximum number of parts in F_i.

So the control on the level of inventory positions with an installation kanban is more restrictive than with an echelon kanban. On the other hand the production may be slowed down by this control on inventory positions.

6.1.2. Identification of properties of some policies

It is also possible to compare a policy with itself with different parameters values in order to find some properties. That way we can find some useless kanbans for the GKCS as it was pointed out in (Buzacott 1989).

As an illustration, we consider again the installation kanban and echelon kanban policies presented in section 5.3.1 and we use our formulation to find how deadlocks can be avoided:

In algorithm 5.2 the system is considered to be not alive if there exists i and j such that $\left\lfloor C_{(i,j)} \right\rfloor_{Q_i}$ is equal to zero.

With the installation kanban we have (see equation(19)):

$$\left\lfloor C_{(i,j)} \right\rfloor_{Q_i} \geqslant \left\lfloor C_{(i,i)} \right\rfloor_{Q_i} = \left\lfloor K_i^{\mathrm{I}} Q_i \right\rfloor_{Q_i} \qquad \text{for } 1 \leqslant i \leqslant j \leqslant N.$$

Then a deadlock is avoided if $\left\lfloor C_{(i,i)} \right\rfloor_{Q_i} \neq 0$, which leads to:

$$K_i^{\mathrm{I}} \geqslant 1 \quad \text{for } i = 1, \cdots, N.$$

With the echelon kanban we have (see equation(21)):

$$\left\lfloor C_{(i,N)} \right\rfloor_{Q_i} = \left\lfloor K_i^{\mathrm{E}} Q_i \right\rfloor_{Q_i} \quad \text{for } i = 1, \cdots, N.$$

Then a deadlock is avoided if

$$K_i^{\mathrm{E}} \geqslant 1 \quad \text{for } i = 1, \cdots, N.$$

6.2. *Quantitative comparison between pull policies*

The idea is to calculate an average cost for a large class of policies in order to facilitate their quantitative comparison. The considered cost increases with the number of parts present in the production system and with the number of demands that are not immediately satisfied and are thus backordered. It can be calculated from the steady state probabilities for all the states. We thus have to solve a Markov chain representing all the possible changes in the states of a given production system. We show here how this Markov chain can be obtained using the formulation proposed in this paper. In order to limit the complexity of this Markov chain, we limit here our study to the case of a 2-stage system, with exponential service times. The state vector is then $\vec{X} = (x_1, x_2)$ and enables to count the number of finished parts (x_2, when it is positive), the number of backordered demands (x_2, when it is negative) and the number of intermediate products

in the production system (x_1). In each stage i, the exponential station with service rate μ_i is stopped when $f_i(\vec{X}) = 0$ or working, with rate μ_i, when $f_i(\vec{X}) \neq 0$. Demands arrive according to a Poisson process with rate λ. There is an infinite number of states $\vec{X}$ since there is not limit to the accumulation of backordered demands (x_2 can thus tend towards $-\infty$). The possible states and transitions depend on $f_1(\vec{X})$ and $f_2(\vec{X})$, which define the control policy. We limit our study to the policies for which the number x_1 of products in the intermediate buffer and in the manufacturing process of station 2 is bounded by a value called $x_1^{\max}$ (see Figure 10 for an example). This enables a repetitive structure for the Markov chain. We also assume that the control curve of station 2 consists in producing only until a level $x_2^{\max}$ of finished products is reached in the output buffer of station 2. We thus have $f_2(\vec{X}) = min(x_1, x_2^{\max} - x_2)$. At $x_2^{\max}$, the control curve of station 2 has a given abscissa, x_1 which increases as x_2 decreases. When it reaches the value $x_1^{\max}$, the control curve of station 1 becomes vertical and authorizes the processing of at most $x_1^{\max}$ products. As its structure is repetitive and infinite, the Markov chain can be solved using the matrix geometric method (Neuts 1981). In the next section, we propose a way to number the states in order to obtain a Markov chain that can be solved using the matrix geometric method. Note that the base stock policy can not be studied the same way since the structure of the obtained Markov chain is not repetitive, due to a non bounded value of x_1.

6.2.1. States numeration

Figure 10 illustrates the proposed states numeration for a policy such that $x_1^{\max} = 5$. This numeration counts the states from the left to the right, and from the top to the bottom. We can observe an horizontal line, between ordinates 4 and 5, which represents the control curve for station 2. We have $x_2^{\max} = 5$. The numbers in grey represent the states for which both stations are on. For state 1, station 1 is on and station 2 is off. For state 2, both stations are off. For state 5, station 2 is on and station 1 is off. The transitions between states are shown on the right part of Figure 10.

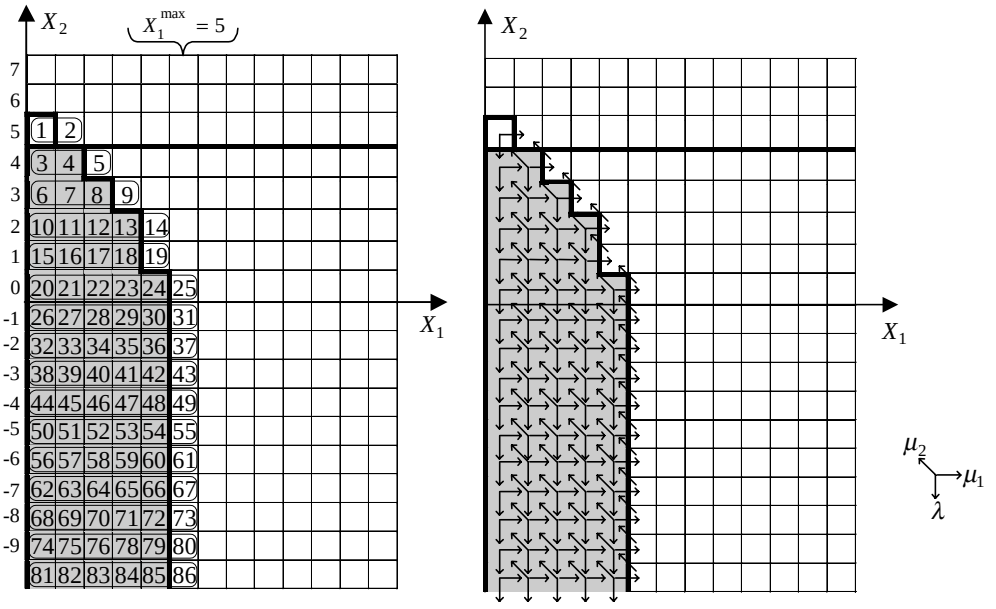


Figure 10. An example of states numeration and transitions between states for a policy

Let us consider the particular case of a generalized kanban system.

Figure 11 illustrates the control curve and the infinitesimal generator for a generalized kanban system with parameters: $K_2 = 1$, $S_1 = 1$, and $S_2 = 5$ (K_1 taking any value). For this example, we have $x_2^{\max} = 5$ and $x_1^{\max} = 2$.

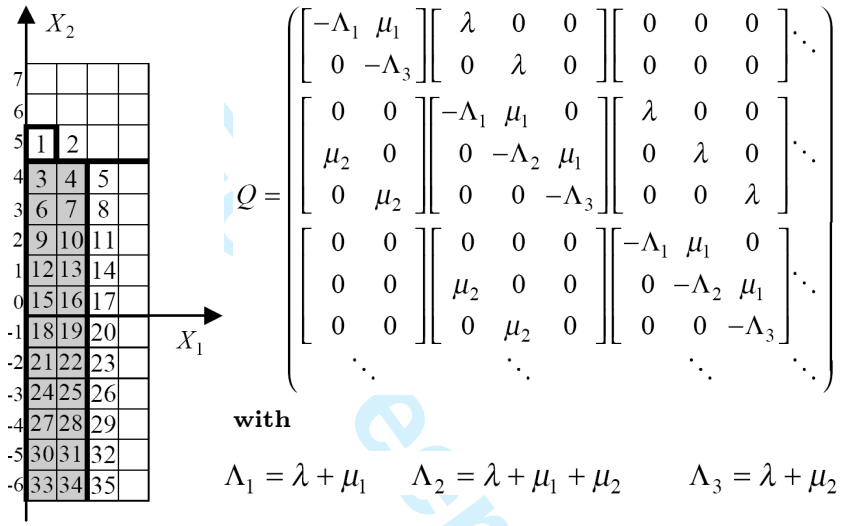


Figure 11. Control curve and infinitesimal generator for a generalized kanban system with $K_2 = 1$, $S_1 = 1$, and $S_2 = 5$.

The structure of the infinitesimal generator shown in figure 11, can be generalized for any policy for which the control curve of station 2 is horizontal and the control curve of station 1 becomes vertical after a given value of x_2 (see figure 10). We then obtain an infinitesimal generator with the following structure:

$$Q = \begin{pmatrix} B_0 & A_0 & 0 & 0 & 0 & 0 & \ddots \\ C_0 & B_1 & A_1 & 0 & 0 & 0 & \ddots \\ 0 & C_1 & \ddots & \ddots & 0 & 0 & \ddots \\ 0 & 0 & \ddots & B_i & A_i & 0 & \ddots \\ 0 & 0 & 0 & C_i & B_i & A_i & \ddots \\ 0 & 0 & 0 & 0 & C_i & B_i & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \tag{22}$$

The rank i used in matrix (22) represents the number of ordinates x_2 for which there are less than $(x_1^{\max} + 1)$ states, i.e. the number of lines, among the states, for which the control curve is not yet vertical.

On figure 10, for x_2 varying from 5 to 1 there are less than 6 states by line, thus $i = 5$. On figure 11, there are less than 3 states by line only for x_2=5, and, again, $i = 1$.

Components A_0, A_1, ...A_i, B_0, B_1, ...B_i, C_0, C_1, ... and C_i are submatrices. For $j < i$ submatrices A_j, B_j and C_j have respectively n, n and m rows and m, n and n columns.

Number n (respectively m) corresponds to the number of states whose ordinate x_2 is $(x_2^{\max} - j)$ (respectively $(x_2^{\max} - j - 1)$). If $n = m$ then sub-matrices A_j, B_j, and C_j have components that are identical to those of A_i, B_i, and C_i, but they have less rows and colums. Submatrices A_i,B_i, and C_i have $x_1^{\max}$ rows and $x_1^{\max}$ columns; they are defined by:

$$C_i = \begin{pmatrix} 0 & & & \\ \mu_2 & 0 & & \\ & \mu_2 & \ddots & \\ & & \ddots & 0 \\ & & & \mu_2 \; 0 \end{pmatrix}, \; B_i = \begin{pmatrix} -\Lambda_1 & \mu_1 & & & \\ & -\Lambda_2 & \ddots & & \\ & & \ddots & \mu_1 & \\ & & & -\Lambda_2 & \mu_1 \\ & & & & -\Lambda_3 \end{pmatrix}, \text{ and } \; A_i = \begin{pmatrix} \lambda & & & & \\ & \lambda & & & \\ & & \ddots & & \\ & & & \lambda & \\ & & & & \lambda \end{pmatrix}$$

with $\Lambda_1 = \mu_1 + \lambda$, $\Lambda_2 = \mu_1 + \mu_2 + \lambda$, and $\Lambda_3 = \mu_2 + \lambda$.

Each submatrix represents transitions between states subsets; these subsets contain states having the same ordinate x_2 in the state space.

These subsets are represented in figure 10 by ovals.

Assuming that $f_2\left(\vec{X}\right) = \min\left(x_1, x_2^{\max} - x_2\right)$, i.e. station 2 is on as long as a level $x_2^{\max}$ of finished products is not reached, and assuming that $f_1(\vec{X})$ verifies the properties of optimal control curves (i.e. D-monotony), then, we have $n = m - 1$ if $n \neq m$. For $n = m - 1$, B_j has the same structure as B_i, A_j has the same structure as A_i to which the last row is removed, and C_j has the same structure as C_i to which the first row is removed. Submatrix B_0 differs from B_i by its diagonal. Its structure is the following:

$$B_0 = \begin{pmatrix} -\Lambda_1 & \mu_1 & & \\ & \ddots & \ddots & \\ & & -\Lambda_1 & \mu_1 \\ & & & -\lambda \end{pmatrix} \tag{23}$$

With the above infinitesimal generator, we are now able to use the matrix geometric method (Neuts 1981) in order to calculate the steady state probabilities for all the states. From these probabilities, we can derive the average number of finished products, $\bar{x}_2^+$, the average number of backordered demands, $\bar{x}_2^-$, and the average number of intermediate products in the production system, $\bar{x}_1$, by multiplying each possible value of variables x_2^+, x_2^-, and x_1 by its probability. Then, we obtain the average cost as a linear combination of these values.

7. Conclusion

In this paper, we showed that it is possible to describe a very large class of pull control policies (including well known ones, like kanban, CONWIP, basestock, generalized kanban, extended kanban, but also many other hybrid policies, and their extensions to systems producing batches) thanks to a rather simple mathematical formulation. This formulation consists in a set of functions, which is the same for all the studied policies, and whose parameters differ from one policy to another.

We also provided computing algorithms that enable to calculate these parameter values for different categories of systems modeled by queueing networks. These algorithms rely on the shortest path search and on the use of $(\min, +)$ algebra tools, that facilitate calculation and enable us to obtain concise expressions. They can easily be extended to other production systems, like for example systems with assembly (or disassembly) between stages.

The simplicity of our formulation allows to identify the existing equivalences between some policies, by comparing their formulation's parameters. This has been illustrated in (Bollon *et al.* 2004) with the comparison of the dynamics of the extended kanban system and the generalized kanban system. It would be interesting to develop an algorithm allowing a systematic search of all the identical dynamics between two systems. Thus, when introducing new control policies, this would facilitate the comparison of their behavior to the behavior of existing ones.

Finally, this formulation can also be useful for the derivation of methods for evaluating and comparing the performance of several pull control policies. We showed how a Markov Chain can be obtained from the canonical formulation of several pull control policies and solved in order to calculate the average cost associated with a policy. For simple cases, limited to two stages with exponential service time, we are able to derive the exact steady-state probabilities. It would be interesting to find a unique method enabling the analysis of most of the pull control policies existing in the literature with more than two stages. Such a common method would simplify the quantitative comparison of the pull policies.

Acknowledgement

We are grateful to the anonymous referees for their careful reading and constructive comments, which have improved the presentation of the paper.

Appendix A. Appendix

A.1. *Properties for the operator* $\lfloor \cdot \rfloor_Q$

Let $\lfloor \alpha \rfloor$ be the highest integer smaller than or equal to α and $\lfloor \alpha \rfloor_\beta = \beta \left\lfloor \frac{\alpha}{\beta} \right\rfloor$ be the highest multiple of β smaller or equal to α. We derived the following properties, for a and b in $\mathbb{R}_{\min}$, and c and d in $\mathbb{N}$, that will be useful in the proof below:

$$\lfloor a \oplus b \rfloor_c = \lfloor a \rfloor_c \oplus \lfloor b \rfloor_c \tag{A1}$$

$$\lfloor a \otimes \lfloor b \rfloor_c \rfloor_{cd} = \lfloor \lfloor a \rfloor_c \otimes \lfloor b \rfloor_c \rfloor_{cd} = \lfloor \lfloor a \rfloor_c \otimes b \rfloor_{cd} \tag{A2}$$

$$\lfloor a \otimes \lfloor b \rfloor_{cd} \rfloor_c = \lfloor a \rfloor_c \otimes \lfloor b \rfloor_{cd} = \lfloor \lfloor a \rfloor_c \otimes \lfloor b \rfloor_{cd} \rfloor_c . \tag{A3}$$

If $\vec{Q}$ and $\vec{X}$ are vectors, respectively in $\mathbb{N}^n$ and in $\mathbb{R}^n_{\min}$, then we define another operator $\left\lfloor \vec{X} \right\rfloor_{\vec{Q}}$ which is in $\mathbb{R}^n_{\min}$, and whose component i is given by $\lfloor x_i \rfloor_{q_i}$.

A.2. *Proof of algorithm 5.2*

Counters $u_i(t)$, $v_{i-1}(t)$, and $w_{(i,j)}(t)$ are batch counters, so if we want to count parts it suffices to multiply them by Q_i, which gives respectively, with $(\min,+)$ algebra notations, $u_i^{Q_i}(t)$, $v_{i-1}^{Q_i}(t)$, and $w_{(i,j)}^{Q_i}(t)$. We still consider that Q_i is a multiple of Q_j whenever $i<j$.

The presence of these batches modifies equation 4. Consider for example a synchronization station U which delivers batches of size Q, queues R and S are linked on to U and they are respectively fed with V and W which respectively deliver batches of size Q' and Q". The value for u^Q is then the minimal solution of the following $(\min,+)$ inequality:

$$u^Q(t) \succeq \lfloor r(0)\, v^{Q'}(t) \oplus s(0)\, w^{Q"}(t) \rfloor_Q .$$

For the queueing system described in proposition 5.1 we can obtain a system of inequalities. Let n_i be the number of synchronization stations V_{j-1} and $\mathrm{W}_{(j,k)}$ for $1 \leqslant j \leqslant i \leqslant N$. Let $\vec{X}(t)$ be a vector of $\mathbb{R}_{\min}^{(1+n_{N+1})}$ whose components are:

$$x_{n_i+1}(t) = v_i^{Q_{i+1}}(t)\,,\; x_{n_i+j}(t) = w_{(i+1,j-1)}^{Q_{i+1}}(t) \text{ and } x_{1+n_{N+1}}(t) = d(t)\,.$$

Let $\vec{Y}(t)$ be a vector of $\mathbb{R}_{\min}^{N}$ whose components are:

$$y_i(t) = u_i^{Q_i}(t)\,.$$

Let $\vec{\tilde{Q}}$ be a vector defined in $\mathbb{N}^{1+n_{N+1}}$ whose components are $\tilde{q}_j = Q_i$ for $n_{i-1} < j \leq n_i$. We then obtain

$$\vec{X}(t) \succeq \left\lfloor A\,\vec{X}(t) \oplus B\,\vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}}, \tag{A4}$$

where A and B are matrices which belong respectively to $\mathbb{R}_{\min}^{(1+n_{N+1})\times(1+n_{N+1})}$ and $\mathbb{R}_{\min}^{(1+n_{N+1})\times N}$. The operator $\lfloor\cdot\rfloor_{\vec{\tilde{Q}}}$ is defined at the end of appendix section A.1. We can also express the inequalities system (A4) as below:

$$x_i(t) \succeq \left\lfloor \bigoplus_{j=1}^{1+n_{N+1}} a_{ij}\, x_j(t) \oplus \bigoplus_{j=1}^{N} b_{ij}\, y_j(t) \right\rfloor_{\tilde{q}_i} \quad \text{for } i = 1 \text{ to } (1+n_{N+1}). \tag{A5}$$

If $i<j$, there is no arc from V_{i-1} or $\mathrm{W}_{(i,k)}$ to V_j or $\mathrm{W}_{(j,l)}$, and then we obtain:

$$j \leq n_k < i \quad \Rightarrow \quad a_{ij} = \varepsilon. \tag{A6}$$

Between U_j and V_j there is an arc going into the queue P_j, so $b_{ij} = P_j$ for $i = n_j + 1$. Also, if $i<j$, there is no arc from U_i to V_j or $\mathrm{W}_{(j,k)}$. So we have

$$(i > n_j + 1) \quad \Rightarrow \quad b_{ij} = \varepsilon. \tag{A7}$$

We are now going to prove that the minimal solution of the system (A4) is

$$\vec{X}(t) = \left\lfloor \tilde{A}^*\, \tilde{B}\, \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}}, \tag{A8}$$

where $\tilde{A}$ and $\tilde{B}$ are matrices which belong respectively to $\mathbb{R}_{\min}^{(1+n_{N+1})\times(1+n_{N+1})}$ and $\mathbb{R}_{\min}^{(1+n_{N+1})\times N}$; the components of $\tilde{A}$ are $\tilde{a}_{ij} = \lfloor a_{ij} \rfloor_{\tilde{q}_j}$ and those of $\tilde{B}$ are $\tilde{b}_{ij} = \lfloor b_{ij} \rfloor_{Q_j}$ if $i \leq n_j$, and for $i > n_j$ we have $\tilde{b}_{ij} = \lfloor b_{ij} \rfloor_{\tilde{q}_i}$ (i.e. $\tilde{b}_{ij} = \lfloor b_{ij} \rfloor_{\min(\tilde{q}_i, Q_j)}$). The star operator is defined as follows:

$$\tilde{A}^* = \bigoplus_{k \geqslant 0} \tilde{A}^k. \tag{A9}$$

From equation (A8), equation (A4) and the properties given by equations (A1),(A2),(A3). we obtain:

$$\begin{aligned} \vec{X}(t) &\succeq \left\lfloor A \left\lfloor \tilde{A}^* \tilde{B} \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}} \oplus B \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}} \\ &\succeq \bigoplus_{k \geqslant 0} \left\lfloor A \left\lfloor \tilde{A}^k \tilde{B} \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}} \right\rfloor_{\vec{\tilde{Q}}} \oplus \left\lfloor B \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}}. \end{aligned} \tag{A10}$$

For $\vec{C}$ a vector in $\mathbb{R}_{\min}^{(1+n_{N+1})}$ we obtain from equations (A2) and (A6):

$$\left\lfloor A \left\lfloor \vec{C} \right\rfloor_{\vec{\tilde{Q}}} \right\rfloor_{\vec{\tilde{Q}}} = \left(\left\lfloor \bigoplus_{j=1}^{1+n_{N+1}} a_{ij} \lfloor c_j \rfloor_{\tilde{q}_j} \right\rfloor_{\tilde{q}_i} \right)_i = \left(\left\lfloor \bigoplus_{j=1}^{1+n_{N+1}} \lfloor a_{ij} \rfloor_{\tilde{q}_j} c_j \right\rfloor_{\tilde{q}_i} \right)_i = \left\lfloor \tilde{A} \vec{C} \right\rfloor_{\vec{\tilde{Q}}},$$

so we have

$$\left\lfloor A \left\lfloor \tilde{A}^k \tilde{B} \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}} \right\rfloor_{\vec{\tilde{Q}}} = \left\lfloor \tilde{A}^{k+1} \tilde{B} \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}}. \tag{A11}$$

From equations (A2), (A3) and (A7) we also have the property:

$$\begin{aligned} \left\lfloor B \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}} = \left\lfloor B \left\lfloor \vec{Y}(t) \right\rfloor_{\vec{Q}} \right\rfloor_{\vec{\tilde{Q}}} &= \left(\left\lfloor \bigoplus_{j=1}^{N} b_{ij} \lfloor y_j(t) \rfloor_{Q_j} \right\rfloor_{\tilde{q}_i} \right)_i \\ &= \left(\left\lfloor \bigoplus_{j=1}^{N} \lfloor b_{ij} \rfloor_{\min(\tilde{q}_i, Q_j)} \lfloor y_j(t) \rfloor_{Q_j} \right\rfloor_{\tilde{q}_i} \right)_i = \left\lfloor \tilde{B} \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}}. \end{aligned} \tag{A12}$$

Then from equations (A10), (A11) and (A12), we can obtain[1] as shown in (Baccelli *et al.* 1992) for the solution (5), that $\vec{X}(t) = \left\lfloor \tilde{A}^* \tilde{B} \vec{Y}(t) \right\rfloor_{\vec{\tilde{Q}}}$. The functions $f_i(\vec{X})$ are obtained with $\frac{x_{n_{i-1}+1}(t)}{y_i(t)} = \frac{v_{i-1}^{Q_i}(t)}{u_i^{Q_i}(t)} = f_i(t)$ and from equations (A13) similar to equations (9):

[1]The set $\mathbb{R}_{\min}$ has to be completed with $(-\infty)$, it is then denoted by $\overline{\mathbb{R}}_{\min}$. It is assumed that $(-\infty)+(+\infty) = +\infty$.

$$\frac{u_i^{Q_i}(t)}{u_{i+1}^{Q_{i+1}}(t)} = \frac{\lfloor x_i(t)\rfloor_{Q_{i+1}}}{\lfloor x_i(0)\rfloor_{Q_{i+1}}} = \frac{\lfloor x_i(t)\rfloor_{Q_{i+1}}}{\lfloor p_i(0)\rfloor_{Q_{i+1}}} \quad \text{and}$$
$$\frac{u_N^{Q_N}(t)}{d(t)} = \frac{x_N(t)}{x_N(0)} = \frac{d_N(0)}{p_N(0)} \otimes x_N(t). \tag{A13}$$

We then prove algorithm 5.2 as it is done for algorithm 4.2.

References

Axster, S. and Rosling, K., 1993. Installation vs. echelon stock policies for multilevel inventory control. *Management Science*, 39 (10), 1274–1280.

Baccelli, F., *et al.*, 1992. *Synchronization and Linearity : An Algebra for Discrete Event Systems*. New York: John Wiley and Sons.

Bollon, J.M., 2001. Etude de différentes politiques de pilotage de systèmes de production. Thesis (PhD). Institut National Polytechnique de Grenoble, LAG.

Bollon, J.M., Di Mascolo, M., and Frein, Y., 2004. Unified framework for describing and comparing the dynamics of pull control policies. *Annals of Operation Research, special issue on Stochastic Models of Production-Inventory Systems*, 125, 21–45.

Bonvik, A., 1996. Performance Analysis of Manufacturing Systems Under Hybrid Control Policies. Thesis (PhD). Massachusetts Institute of Technology.

Bonvik, A., Couch, C., and Gershwin, S., 1996. A comparison of production line control mechanisms. *International Journal of Production Research*, 35, 789–804.

Boonlertvanich, K., April 2005. Extended CONWIP-Kanban system: control and performance analysis. Thesis (PhD). Georgia Institute of Technology.

Buzacott, J., 1989. Queueing models of kanban and MRP controlled manufacturing Systems. *Engineering Cost and Production Economics*, 17, 3–20.

Dallery, Y. and Liberopoulos, G., 2000. Extended kanban control system: Combining kanban and base stock. *IIE Transactions on Design and Manufacturing*, 32 (4), 369–386.

Duri, C., Frein, Y., and Di Mascolo, M., 2000. Comparison among three pull control policies: kanban, base stock, and generalized kanban. *Annals of Operations Research*, 93, 41–69.

Geraghty, J. and Heavey, C., 2005. A review and comparison of hybrid and pull-type production control strategies. *OR Spectrum*, 27, 435 – 457.

Kleijnen, J. and Gaury, E., 2003. Short-term robustness of production management systems: A case study. *European Journal of Operational Research*, 148, 452 – 465.

Lee, Y. and Zipkin, P., 1992. Tandem queues with a planned inventories. *Operations Research*, 40 (5), 936–947.

Liberopoulos, G. and Dallery, Y., 2000. A unified framework for pull control mechanisms in multi-stage manufacturing systems. *Annals of Operations Research*, 93, 325–355.

Liberopoulos, G. and Dallery, Y., 2003. Comparative modelling of multi-stage production-inventory control policies with lot sizing. *International Journal of Production Research*, 41 (6), 1273–1298.

Monden, Y., 1983. *Toyota Production system*. Norcross, Georgia, Industrial Engineering and Management Press (Institute of Industrial Engineers).

Neuts, M., 1981. *Matrix-geometric solutions in stochastics models : an algorithmic approach*. Londres, R.U.: The John Hopkins University Press.

Spearman, M., Woodruff, D., and Hopp, W., 1990. CONWIP: A pull alternative to kanban.. *International Journal of Production Research*, 28 (5), 879 – 894.

Veatch, M. and Wein, L., 1994. Optimal control of a two-station tandem production/inventory system. *Operations Research*, 42 (2), 337–350.